2011/2012 CM50109 Group Project - Airline Reservation System

Group Submission due in: 4 pm Wednesday 12th December 2012
Individual Submission due in: 4pm Monday 7th January 2013

This project will enable you to gain experience in:

- group working;
- the software lifecycle for solving a substantial software problem;
- developing an awareness of the strengths and limitations of the implementation language;
- using appropriate software development techniques for design, implementation and testing;
- writing software development, software maintenance and user documentation;
- evaluating all aspects of the development process and the final product.

1 User Requirements

A computerised airline reservation system with the following properties:

- The system must be able to read in two groups of input data:
  - A flight control group containing flight numbers, seating capacities, and flight dates (day, month and year only, not time of day). This information must be read in from a text file.
  - Assorted requests to be entered interactively.
- The system must handle four basic types of request:
  - A reservation
  - A cancellation
  - A passenger inquiry
  - A flight inquiry

  Reservation: specifies a passenger name, a flight number and the class of seat required (First Class or Economy class — for any flight, assume that 10% of the total seats on the flight are First Class, and the rest are Economy). The passenger is placed on the passenger list for that flight - if there is room available in the appropriate class. If not, the passenger should be placed on the end of the waiting queue for the flight (again for the appropriate class). However, a passenger requesting a first class seat, on a flight where the first class seats are all full, should be offered an economy class seat if any are available for the flight.

If the economy seats are also full, a passenger, whose initial request was for a first class seat, should be allowed to wait on either the First Class or Economy waiting lists for the flight (but not on both). A passenger may appear on the passenger lists for more than one flight, but is not allowed to be booked more than once on a single flight, or on more than one flight per day. A message should be given to the operator indicating what action has been taken.

Cancellation: specifies a passenger name and a flight number. The passenger is removed from the passenger list or waiting list of the flight (and class) in question. The first passenger on the appropriate waiting list should be promoted onto the passenger list. Again a message is required showing what action has occurred.

Passenger Inquiry: Specifies a passenger name only. This should print a list of all the flights on which the passenger is booked or waiting (indicating the status and the seat class for each).

Flight Inquiry: Specifies a flight number. This should print the passenger and waiting lists for the given flight, and the date on which it occurs.

The flight control group file (filename: `FCGDATA`) contains information consisting of flight number, number of seats, and time (in 24 hour clock format) and date of flight (ddmmyyyy) in the following format:

```
BA679 250 0950 25091998
TW098 10 1005 01042000
AA321 380 0800 31121998
```

Note that the number of spaces between the fields in this file can vary. So, for example, the following flight control groups and requests should be allowed by the system:

```
       BA679 250     0950      25091998
TW098    10 1005         01042000
AA321 380                  0800 31121998
```

- The system must be as robust as possible. For example, the system must recognise errors in the input files (and user inputs), and act appropriately to deal with them, though exactly how these are dealt with is left to the developers.
- The user interface to the reservation system should be as "user-friendly" as possible, to cope with users who are not computer literate. The user interface can be via command line entry or implemented via a web browser. But keep it simple and clear!
- Each group will be expected to demonstrate their program to me using the PCs in the MSc lab.

2 Coursework Requirements

Your task is to analyse, specify, design and implement the airline reservation system described in the User Requirements given above.

Professor ffitch will allocate students to their project groups.

Final Project hand-in dates

Group Submission: 4 pm Wednesday 12th December 2012 via the Department. This deadline is fixed. Late work will lose marks.

Individual Submission: 4pm Monday 7th January 2013. You can email it to me (as a pdf) if you are not back at the University for the hand-in, but you MUST PROVIDE A PRINTED COPY when you arrive back at the University for your exams.

2.1 Group Submission (85% of your mark comes from this)

As a group you are required to hand in two documents for the group project. Both documents must be wire spiral bound, have a transparent (acetate) front cover and card back cover, and be printed double-sided. The documents must contain the following information:

Document One

- A front page identifying the group and the group members.
- A second page that is the (completed) group coursework submission sheet.
- A Project Overview. This should be very a short section at the start of this document which ties together all the other documents, indicates clearly how the work was divided up amongst the individual group members, and covers any additional aspects of the project that the group wishes to report.
- A complete requirements analysis and resulting requirements specification for the problem.
- An analysis of how to solve the problem, including a modularisation showing how you determined the various abstractions required in the problem solution and software design, including, where possible, formal specifications of the abstractions you have determined;
- A complete design of the software system which uses the abstractions to implement a problem solution that meets the requirements. This will include designs of individual classes, methods etc. Remember to design the overall structure of the software system, as well as the individual parts. Make sure it is clear which class implements which abstraction(s).

  You should show the overall structure of the entire software system, through the individual classes, the methods in each class, down to the level of individual algorithms. It is a good idea to provide a diagram showing how the various system parts relate to each other. Remember to relate these implementation-level components back to the abstractions in your modularisation.

- A draft User Guide, describing what the system does, and how to use it (including installation instructions). Keep it brief but informative.

Document Two

- A front page identifying the group and the group members.
- A second page that is the (completed) group coursework submission sheet.
- Test plans and test results for the system. You must create your own FCGDATA and REQUESTS files, to the specification given above in the user requirements. They will not be provided for you. Note that these two files must be human-readable (ie. text files). It is up to you to develop appropriate test plans and test data for all the levels of testing that you require. You are unlikely to have time to perform full testing. Show some key black box testing, and one or two examples of white box testing.
- A draft Maintenance/Programmers Guide which details the key issues in the implementation for anyone who may have to maintain your program. It should clearly describe what functionality is found in each class. Remember that this guide is for the benefit of someone who has not had any part in writing the code but may have to change it later. Keep it brief but informative.
- A fully commented implementation (in C++) of the reservation system. This must be reduced in size, and printed out as a least two pages of code per A4 page.
- A disk or CD-ROM containing the completed system. This should include the uncompiled code, and a ready-to-run version of the system, and the group documents. Plus any extra data files it may need to use. You will be required to give a demonstration of your running system.
- A Project Diary from each member of the group, detailing your day-to-day work on the project. This should be a clear and truthful indication of what each of you did for the project on each day for the whole of the project period.
- Minutes from all the group meetings.

Note: Missing sections in either document will result in loss of marks, as will binding sections into the wrong document. I will not accept any more than TWO documents (plus questionnaires) from each group.

2.2 Individual Submission (15% of your mark comes from this)

Each member of the group must hand in an individual submission. The individual submission must consist of:

- A Project Critique. This must be a detailed review of the system, considering both the final product and the development process. Highlight how you would now do things differently, what additional functionality you would include and, where appropriate, an outline of how it might be implemented. Discuss the implementation language, its ease of use, limitations, etc.

- A discussion of the legal and ethical issues associated with the use of the system, with consideration of how these issues would affect the design and implementation of a commercial version of the system.

The individual submission must be your own work. You must not collaborate with your group colleagues to write this document. The individual submission should be about five sides of A4 in a 10 or 11 point font. The individual submissions should be handed in separately to the group submission (note that there is a different hand-in date for these).

2.3 Group and Individual Questionnaires

In addition, you must hand in, at the same time as your group documents (12th December 2011):

- A Group Questionnaire (which will be provided). One per group, filled in by the group as a whole. On this you indicate the group's view of how much work each group member did and what the group thinks their individual marks should be as a percentage of the project. This must be signed by all members of the group. Any student who does not agree with the groups view of their individual mark should see John ffitch immediately after the project hand-in date. Do not bind this into your project documents, but hand it in separately.
- A confidential Personal Questionnaire (which will be provided), one per group member, describing your individual views on how the project went for you. This should be handed in sealed in an envelope.

3 Project Aims and Advice

The aim is not just to design and build software which meets the given requirements, but also to learn something about working in a group and building a large piece of software.

I expect you to use the project to put into practice good software engineering techniques. On the design side these include — formal specification, modularisation, data abstraction etc. On the implementation side they include — library classes, software testing (also part of design), dynamic implementations of data abstractions etc.

The description given at the top of this sheet is the user requirements, and needs to be tightened up, and any ambiguities removed. Note that the user requirements are not negotiable – the system **must** fulfill these requirements. You may, however, add extra requirements if your requirements analysis shows them to be necessary.

You should plan the modularisation of the project at the beginning, including how it divides up amongst the group members. The analysis, specification, design and test plans should be complete before any implementation starts. You should produce a timescale for the project, including estimates of time and effort for each task, and milestones of important tasks which must be completed before another stage is started. This must be done at the start of the project, so that you can respond to the question "Is the project on target?", at any time. You should also keep track of how accurate your estimates were for each stage.

As part of your design you should use an object oriented design technique (ie modularisation) to find the objects (ie data abstractions) in the system, and, as far as possible, formally specify these objects. Your design should also show the relationships between these objects.

Your group should choose one member to be the group leader. The leader's job is to keep track of the overall organisation of the project, make sure everyone gets a fair amount of work to do, make sure everyone pulls their weight, and generally manage the overall structure of the project. It's a good idea to choose someone who is a good planner, and quite methodical for the group leader. The leader isn't necessarily the person in the group with the most programming experience. Note, the leader will also have to do some of the design/ implementation/ documentation etc, just like the rest of the group, although allowances will be made for the time spent managing the group. Things to consider:

- How best to divide the work up between the members of the group. This is an exercise in project management as well as software development. Plan your time carefully. CM50109 is a double unit, and this group project is worth 60% of the mark for the unit. You will also have two individual courseworks for the other 40%.

  Important: Each member of the group should take on some of the analysis, design, programming, documentation etc. This is perhaps not very realistic, since in industry it is more usual for different people to do the design and programming, but it is necessary in this context so that all group members have an equal opportunity to experience the various stages in software development.

- The objects (data abstractions) in the problem — these will provide you with the class decomposition of the problem, and thus an idea of how to divide up the work amongst yourselves.

- The user interface. What is a "good" user interface? How do you make it "user-friendly"? How do you make it robust? User interfaces do not need to be complex and highly graphical to be "good".

- Suitable data structures for storing flight control information. You could decide not to work directly from the file, but to read chunks of the information from the file into a data structure as it is needed. Remember, it will have to be searched many times. Suitable data structures for storing the passenger lists and waiting lists for each flight. A flight cross-reference list may be needed for each passenger too.

- Data abstractions, data structures, library classes required – the interfaces should be defined and finalised before the group splits up to design and implement their separate parts, otherwise you will have trouble integrating the various parts of the system.

- Error handling. How will your system deal with errors in, for example, the FCGDATA and REQUESTS files (eg. incorrect, missing or incomplete data fields)? How will the user interface deal with incorrect or inappropriate user input.

The above list is not an exhaustive list of things you need to consider when designing and implementing your system; there are lots of other areas that need thinking about. There are many good books available on software engineering.

4 Assignment of Marks

For each individual, your final mark is made up as follows: 85% from the group hand-ins, as specified above, and 15% from your individual project critique.

Normally each individual member of the group will receive the same group mark. However, in some cases the group component of an individual's mark may be adjusted up or down from the group mark , if, for example, an individual has failed to perform sufficient work, or has produced exceptional work for the project. Questionnaires and Project Diaries may be used to help in the assignment of marks.

Threshold
Single user system.
Group documentation shows appropriate planning, basic understanding of the software lifecycle as applied to the project, with basic requirements, simple design, implementation and testing. The resulting software is fairly functional, and has basic maintenance documentation and basic user guide. Overall the project shows straightforward use of software development techniques. Individual report gives a basic review of the software development and the final product, and a simplistic, but correct consideration of legal and ethical issues, with appropriate, but brief consideration of the design and implementation implications.

Good
Group documentation shows good planning, good understanding of the software lifecycle as applied to the project, with appropriate and effective requirements, good design, implementation and well-thought out and executed testing. Overall the project shows intelligent use of appropriate software development techniques. The resulting software is very functional and has good maintenance documentation and user guide. Individual report gives a thoughtful and detailed review of the software development process and the final product, and a simplistic, but correct consideration of legal and ethical issues, with appropriate, but brief consideration of the design and implementation implications.

Distinction
Group documentation shows excellent planning, excellent understanding of the software lifecycle as applied to the project, with excellent requirements, excellent design, implementation and very well-thought out and executed testing. Overall the project shows intelligent and insightful use of appropriate software development techniques and excellent time management. The resulting software is very functional and has excellent maintenance documentation and user guide. Individual report gives a very

thoughtful, insightful and detailed review of the software development process and final product.
JPff after CPW 13/11/12